UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Enhancing controllability of text generation

Author:
Anton SHCHERBYNA

Supervisor:
Kostiantyn OMELIANCHUK

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

I, Anton SHCHERBYNA, declare that this thesis titled, "Enhancing controllability of text generation" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Think globally, act locally!"

Bohdan Hawrylyshyn

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Enhancing controllability of text generation

by Anton SHCHERBYNA

Abstract

Many models could generate text conditioned on some context, but those approaches don't provide us with the ability to control various aspects of the generated text (e.g., sentiment). To address this problem, Variational Autoencoder is typically used because they give the ability to manipulate in latent space and, in this way, control text generation. However, it has been shown that VAE with strong autoregressive decoders, which are used for text modeling, faces posterior collapse problem. We think that one of the reasons why this problem occurs is a restrictive gaussian assumption we made about approximate posterior. In this work, we want to apply well-known approaches based on Normalizing Flows to improve approximate posterior for text modeling and check if it can help avoid posterior collapse.

Acknowledgements

I want to thank UCU and all its people for their day-to-day contribution to building the most fantastic community in Ukraine. I especially want to thank Olexii Molchanovskyi and the the whole team of APPS faculty for organizing this master's program. Of course, I would like to thank my supervisor Kostiantyn Omelianchuk for proposing the topic of this work and his intellectual guidance. I thank Artem Chernodub for mentorship during seminars and thoughtful questions. Also, I'm thankful to Grammarly for providing computational resources, which helped a lot during experimentation. Finally, I say thank you to my groupmates, friends, and family who were incredibly supportive all this time.

Contents

List of Figures

List of Tables

List of Abbreviations

LSTM	**L**ong **S**hort-**T**erm **M**emory
RNN	**R**ecurrent **N**eural **N**etwork
VAE	**V**ariational **A**utoencoder
NF	**N**ormalizing **F**low
IAF	**I**nverse **A**utoregrssive **F**low
NLL	**N**egative **L**og **L**ikelihood
ELBO	**E**vidence **L**ower **B**ound
KL	**K**ullback-**L**eibler
MLP	**M**ultilayer **P**erceptron

List of Symbols

$\mathbb{E}x$	Expectation
$E_\theta(x)$	Encoder network
$D_\theta(x)$	Decoder network
$\mathbb{D}_{\mathbb{KL}}$	Kullback-Leibler divergence
$x \sim p(x)$	Sampling from distribution
$det\frac{df}{dx}$	Determinant of Jacobian of transformation f

Chapter 1

Introduction

1.1 Text Generation Overview

In recent years there was a significant advancement in the field of text generation. In 2014 sequence to sequence models with LSTM encoder and decoder were proposed (Sutskever et al., 2014). This approach became state-of-the-art in the field and was successfully used for various tasks, e.g., machine translation. However, LSTM networks tend to forget information from the whole sequence, so the next significant improvement - attention mechanism - was proposed (Bahdanau et al., 2014). The main idea of this approach is to provide a decoder with the information from each token from the source sequence directly and score each piece of information by usefulness for the decoder. Finally, a pure attentional model, which is called Transformer, was proposed (Vaswani et al., 2017). Since then, transformer-like models became state-of-the-art methods in text representation learning and text generation. For example, BERT released by Google (Devlin et al., 2018) became the standard for extracting representations from texts, and GPT-2 made by OpenAI became the most powerful tool for text generation. In the case of GPT-2, authors even provided weights only for a small model with limited capabilities. They said that their model is capable of producing such high-quality texts, so they fear somebody can use it to produce realistic fakes.

All those models have a similar structure. Typical text generation model consists of encoder $E_\theta(x)$ and decoder $D_\phi(h)$. Both encoder and decoder can be represented as a deep neural network: LSTM (Sutskever et al., 2014), CNN (Ott et al., 2019), or stacked feed-forward networks, which forms transformer-like model Vaswani et al., 2017. Encoder extracts information from the source sequence $\{x_i\}$ into hidden representations $\{h\}$ and then decoder produces target sequence based on those representations (Figure 1.1). Such models trained end-to-end and use various training signals. For example, we can force the encoder to encode one sentence and decoder to produce the next sentence from the same text. We can use the so-called "hidden language model" approach when our sequence to sequence model is forced to predict intentionally deleted tokens from source sequence. Or we can encode source text and then try to decode it as we do with autoencoders. In all cases, we use classic categorical entropy between distribution predicted by network and true distribution as a loss function.

Also, it's worth noting that right now, transformer-like models outperform old models based on LSTM, but they are harder to train, require much more training data and computational resources, so we'll concentrate on LSTM based models. Moreover, there is no difference between LSTM and transformer-based models in terms of our problem so that we can transfer all methods created for the LSTM model to transformer-like.

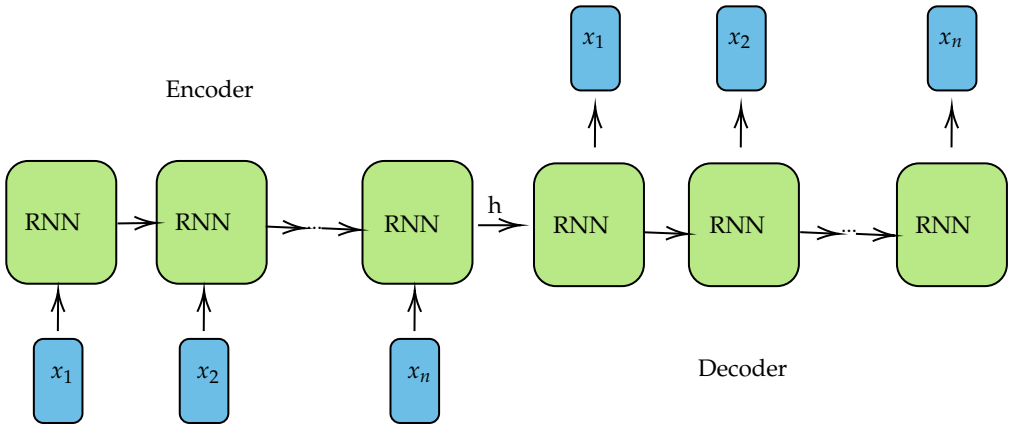

FIGURE 1.1: Simple sequence to sequence model

But all those approaches lack one crucial property - controllability. By controllability, we mean an ability to change attributes of the generated text (e.g., sentiment). Models described above conditioned only on the text they saw previously, which is uninterpretable and unpredictable controllable parameter. Also, the space of hidden representations of such models is unsmooth (Bowman et al., 2015). It means that we can't interpolate in the latent space to discover dependencies between different hidden representations and generated text.

1.2 Variational Autoencoders

There was considerable progress in the direction of the controllable generation in the vision domain. VAE (Kingma et al., 2013) extends classic autoencoder with probabilistic argumentation and gives the ability to control generation by exploring latent space. For this purpose we define latent variable $z \sim p_z(z)$ which has some probabilistic prior distribution (typically Gaussian), then we define some complex conditional distribution $x \sim p_\theta(x|z)$. In the vision domain, it's Normal distribution with mean and variance expressed by a neural network with parameters θ, but in the case of text, it can be categorical distribution over tokens modeled by RNN. Now we can define the likelihood:

$$p_\theta(x) = \int p_\theta(x|z)p(z)dz \tag{1.1}$$

But it appears to be intractable, so we can't optimize it directly. But there is a solution: we introduce new conditional posterior distribution $q_\phi(z|x)$ parameterized by neural network with parameters ϕ (Figure 1.2). Now we can derive a lower bound (ELBO) on the data likelihood $p_\theta(x)$, which is tractable, so we can optimize it with gradient descent. All derivation can be found in (Kingma et al., 2013):

$$\mathcal{L} = \mathbb{E}_{z\sim q_\phi(z|x)} log p_\theta(x|z) - \mathbb{D}_{\mathbb{KL}}(q_\phi(z|x)||p(z)) \leq log_\theta(p|z) \tag{1.2}$$

This loss consists of reconstruction part and Kullback-Leibler divergence between approximate posterior $q_\phi(z|x)$ and prior $p(z)$, so minimizing this term we make approximate posterior closer to model posterior. Using this model, we can encode source sample to latent space, tweak the latent, and decode it.

Also, it's worth to mention, that we can't sample directly from $q_\phi(z|x)$ during training, because in this case, we can't propagate gradients through sampling step,

but we can apply simple reparameterization trick. We can sample from Gaussian distribution $\epsilon \sim N(0, 1)$ and define z as follows:

$$z = \mu + \sigma * \epsilon \tag{1.3}$$

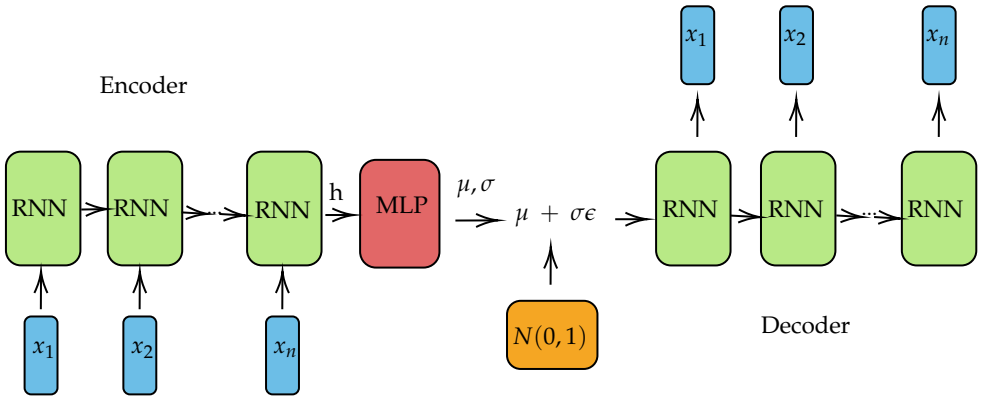

FIGURE 1.2: VAE for text modeling

1.3 Posterior Collapse

However, such approach for controllable data modeling has some drawbacks. The main problem is that the value of KL divergence tends to become zero, which means that our posterior $q_\theta(z|x)$ becomes entirely indistinguishable from prior $p(z)$. One simple explanation of such phenomena is that autoregressive decoder is sensitive to the smallest variations in the hidden state, so a random sampling process from posterior can cause the model to ignore such highly variable values and explain data with the modeling capabilities of the decoder. It causes significant problems when we use the strong autoregressive network as a decoder, which we often do for modeling sequences (in our case - texts). It means that the decoder tends to ignore outputs of the encoder, so we do not backpropagate enough through it, and it produces meaningless latent, which decoder tends to ignore even more (Bowman et al., 2015). This makes our model fully deterministic like a classic autoencoder, and we can no longer use latent to manipulate with the decoder. Solving this problem is crucial for using VAE for controllable text generation.

1.4 Thesis Structure

This thesis has following structure:

1. Chapter 1. We make an overview of the text generation field in general, described the variational approach, which gives an ability to control the generation process and outlined its problem, which we'll try to solve.

2. Chapter 2. We describe and analyze existing approaches to the problem.

3. Chapter 3. We describe the Normalizing Flow framework and how we can apply it to improve the posterior approximation.

4. Chapter 4. We provide details about our experiments and discuss results.

5. Chapter 5. We summarize the outcomes of this work and outline possible directions for future work.

Chapter 2

Related Work

In this section, we'll cover the most recent works in which the posterior collapse problem is faced. It's important to note that we'll cover works that are primarily focused on text modeling with VAE even though this problem is common for other domains, where the strong autoregressive decoder is used. Also, it worth noting that there are exist many more approaches, but those covered in this section are the most common and used as benchmarks in most papers. We'll use them as benchmarks too, so it's important to get the general idea behind each of them and understand their limitations.

2.1 KL-annealing & β-VAE

In the first paper published on this topic (Bowman et al., 2015) authors proposed approach that we described previously and faced problems that we outlined above.

In fact, they were also first to propose a simple solution for the posterior collapse. They added weight to the KL term during training, and at the beginning of the optimization process they set it to zero, so during first steps model can get as much information as it can into variable z, and while training progresses, they gradually increase the weight, forcing approximate posterior to be closer to the prior reducing flexibility for latent variable z.

β-VAE (Higgins et al., 2017) is an extension of a KL-annealing approach, but with deeper optimization explanation. We can represent our ELBO as a constrained optimisation problem:

$$\mathcal{L} = \mathbb{E}_{z\sim q_\phi(z|x)} log p_\theta(x|z), \mathbb{D}_{\mathbb{KL}}(q_\phi(z|x)||p(z)) < \epsilon \tag{2.1}$$

Then using Karush-Khun-Tucker theorem we can introduce Lagrangian:

$$\mathcal{L} = \mathbb{E}_{z\sim q_\phi(z|x)} log p_\theta(x|z) - \beta(\mathbb{D}_{\mathbb{KL}}(q_\phi(z|x)||p(z)) - \epsilon) \tag{2.2}$$

Knowing that $\epsilon \geq 0$ from the properties of KL-divergence and using the same Karush-Khun-Tucker theorem we can rewrite it as:

$$\mathcal{L} = \mathbb{E}_{z\sim q_\phi(z|x)} log p_\theta(x|z) - \beta\mathbb{D}_{\mathbb{KL}}(q_\phi(z|x)||p(z)), \tag{2.3}$$

where β measure the amount of information that z can capture. In the original paper authors claimed that bigger values for β are crucial to learn disentangled representations, as it forces the model to factorize the space more efficiently. Despite interest in disentanglement, to solve the posterior collapse, we can set $\beta \leq 0$ during the whole training process.

However, those approaches have a significant drawback - we optimize not a direct ELBO (in case of a KL-annealing, only during last steps when $\beta = 1$ we do

optimize ELBO). Also, for β-VAE with small β it's hard to tell if the result is better due to this approach or because of a lousy optimization.

2.2 Word Dropout

Another approach to the posterior collapse problem proposed by (Bowman et al., 2015) was word dropout. During training, we decode target sequence not in a purely autoregressive moe, but based on ground-truth samples. In other words, it means that on timestep t, we provide our decoder not token generated by our model on timestep $t-1$, but real token from the data. Such approach helps train decoder faster. As we said previously one of the reasons for the posterior collapse problem is a strong autoregressive decoder, so we can weaken decoder by randomly changing ground-truth tokens with special UNK token, which is used to code unknown words (words that we don't have in a dictionary). However, experiments show that it comes with the cost as reconstruction quality becomes significantly worse.

2.3 Lagging Inference

We can view the problem of posterior collapse from the training dynamics perspective, as presented in (He et al., 2019). Authors of this paper propose a new strategy for training VAE - aggressively optimize encoder. During the classic training step, they update weights ϕ of the encoder for k times instead of one and then update weights of the decoder θ only ones. Authors claim that in such way they can solve a problem (which leads to posterior collapse), that they discover - the approximation of posterior lags behind true posterior in the beginning of the training. This method shows excellent results, but it also increases the time needed for training significantly.

2.4 Semi-Amortized VAE

This work introduces interesting idea based on Stochastic variational inference. To explain this concept, let's view classic VAE from a different perspective. VAE is based on a type of a variational inference, which is called "amortized". It means that variational parameters are predicted by one global inference network for each data sample. This gives the ability to optimize such model end-to-end with decoder and scale to large datasets. However, there exist another approach for variational inference in which we initially set random variational parameters for each data point and then optimize them during training procedure maximizing familiar ELBO instead of optimizing posterior $q_\theta(z|x)$ parametrized by encoder network with parameters θ. So new posterior looks as follows:

$$z \sim q(z; \lambda) \tag{2.4}$$

where $\lambda = [\mu, \sigma]$ is a learnable vector and $q(z; \lambda)$ is still normally distributed. Such approach allows us to obtain much better variational parameters, but the stochastic inference of variational parameter λ for each data sample on each training step is hugely time-consuming. In Semi-Amortized VAE, authors propose to combine both approaches: they initialize variational parameter λ as the output of the encoder network and then refine them using stochastic variational inference. Also, the authors

proposed a way to optimize this model end-to-end propagating through stochastic inference steps. However, the time complexity is still an issue.

2.5 Implicit-VAE

In this work, the authors tried to face both problems. They changed approximated posterior $q_\phi(z|x)$ from Normal distribution with parameters inferred by encoder network with sample-based distribution represented by sampling mechanism based on an encoder network:

$$z = q(x, \epsilon), \tag{2.5}$$

where $\epsilon \sim N(0, 1)$. Also, it's worth noting that they combine ϵ with the last hidden state of the LSTM encoder via MLP. This trick makes approximated posterior much more expressive than Normal distribution with parameters inferred through the encoder. Also, it makes a transition from the last hidden state of the encoder to the initial hidden state of the decoder more smooth, which is essential for solving posterior collapse problem. However, this approach has some optimization problems as KL term with implicit $q_\phi(z|x)$ becomes intractable. Authors proposed to replace KL term with the dual form:

$$\mathbb{D}_{\mathbb{KL}}(q_\phi(z|x)||p(z)) = max_v \mathbb{E}_{z\sim q_\phi(z|x)} v_\psi(x, z) - \mathbb{E}_{z\sim p(z)} exp(v_\psi(x, z)) \tag{2.6}$$

So, in this case, we can't optimize precise ELBO, and also on each training step, we have to evaluate KL term through another optimization procedure, which increases the time needed for training.

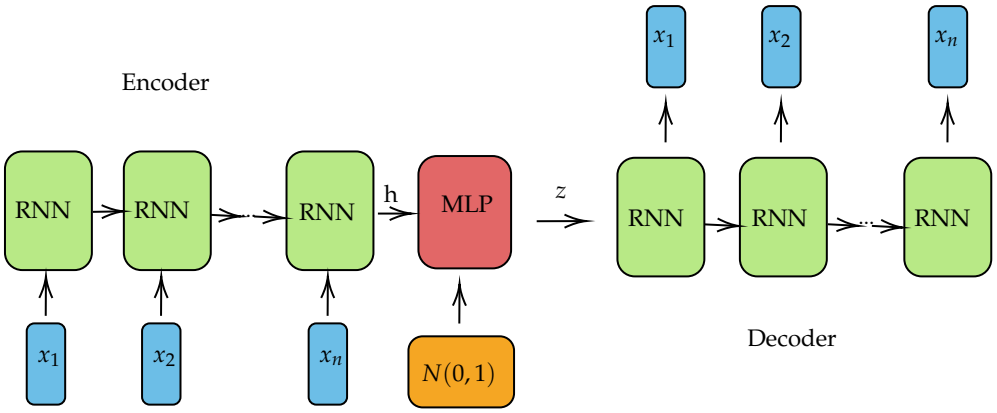

FIGURE 2.1: Implicit-VAE

2.6 Comparison

We aggregated all results 2.1 from the works described above to support our arguments about the drawbacks of each method and provide further thoughts.

As we said, KL-annealing proposed by (Bowman et al., 2015) isn't sufficient to solve the posterior collapse problem as KL value converged to 0. Word dropout technique to weaken decoder helps, but dramatically hurts reconstruction capabilities of the model.

β-VAE shows good results in terms of KL divergence value and data modeling abilities, but we think that it happens due to insufficient optimization during training, so latent produced by this model still useless for the decoder. Approach with aggressive encoder update shows the best results and it suggests that their hypothesis about the reason for the posterior collapse issue could be right.

	Yahoo			Yelp		
Model	NLL	KL	PPL	NLL	KL	PPL
LSTM-LM	328.0	-	-	358.1	-	-
VAE + Annealing	328.6	0.0	-	357.9	0.0	-
β-VAE (β = 0.4)	328.7	6.3	-	358.2	4.2	-
β-VAE (β = 0.8)	328.8	0.0	-	358.1	0.0	-
Word dropout 25%	334.2	1.44	-	-	-	-
Word dropout 50%	345.0	5.29	-	-	-	-
Lagging Inference	328.2	5.6	-	356.9	3.4	-
Lagging Inference + Annealing	**326.7**	**5.7**	**-**	**355.9**	**3.8**	**-**
SA-VAE	329.2	0.1	-	357.8	0.3	-
SA-VAE + Annealing	327.2	5.2	-	355.9	2.8	-
Implicit-VAE + Annealing	309.1	11.4	47.93	348.7	11.6	36.88

TABLE 2.1: Relate work comparison

SA-VAE shows comparable results to Lagging Inference, but as we can see in 2.2, it takes significantly more time to train such a model than Lagging Inference model.

Model	Time
VAE	1
VAE + Lagging inference	2-3
SA-VAE	9-10
Implicit-VAE	1.3

TABLE 2.2: Time spent on training (relative to plain VAE)

Implicit-VAE shows the best performance, but we must remember that they use the estimation of KL divergence, so their results aren't directly comparable to others. It's a bit strange that ELBO, which should consist of the reconstruction part and KL, is much smaller than the likelihood of the data modeled by the pure LSTM model.

To sum up, we can say that Lagging Inference approach with annealing is the most consistent approach with a good trade-off between model quality and training time, interesting view of posterior collapse issue, and most trustful results. So we can use this model as main benchmark for our experiments.

Chapter 3

Normalizing Flows for Variational Inference

In previous section, most of the approaches tried to face posterior collapse problem from different perspectives, but only one tried to face a problem with the perspective of restrictive Gaussian assumption on posterior. However, making posterior more flexible can improve the quality of the whole model. There is some evidence that posterior collapse problem can be caused by a restrictive condition that we put on an approximate posterior (Cremer et al., 2018; Fang et al., 2019) as we model it as a simple Normal distribution with parameters inferred by encoder network, which limits the possibility for approximate posterior to match true posterior distribution. So it's essential to try to face this problem by making posterior more complex and check if it helps. To achieve this, Normalizing Flows framework was proposed in (Jimenez Rezende et al., 2015). Also, an improved autoregressive Normalizing Flow (IAF) for variational inference was proposed in (Kingma et al., 2016). In this section, we'll explain what are Normalizing Flow and autoregressive Normalizing Flow in detail and how they can be used to improve approximated posterior.

3.1 Normalizing Flows

At first we need to recall how we can transform probability density function. Let $z \in R^n$ be a random variable with the density function $q(z)$. Now we define some bijective and differentiable function $f(x) : R^n \to R^n$. Then we can compute density of the transformed variable $\hat{z} = f(z)$ as follows:

$$\hat{q}(\hat{z}) = q(f^{-1}(\hat{z}))|det\frac{df^{-1}}{dz}| = q(f^{-1}(\hat{z}))|det\frac{df}{dz}|^{-1} = q(z)|det\frac{df}{dz}|^{-1} \tag{3.1}$$

where det stands for determinant and the whole term $|det\frac{df}{dz}|$ is a determinant of Jacobian of the transformation. Now we can apply a sequence of such transformations:

$$z_K = f_K \circ ... \circ f_1(z_0), K \in N \tag{3.2}$$

and define density for z_k as:

$$q_k(z_K) = q_0(z_0)\prod_{i=1}^{K}|det\frac{df_k}{dz_{k-1}}|^{-1} \tag{3.3}$$

Such transformation of variable and its density is called Flow and Normalizing Flow, respectively. It's important to choose such transformation f, so its Jacobian is easy to compute, then we can easily use this trick in applications.

3.2 Improve Approximate Posterior with NF

Before we describe examples of such transformations, let's explain how we can apply this approach to improve approximate posterior. Let's recall that in plain VAE our posterior $q_\phi(z|x)$ is defined as Normal distribution with parameters μ_ϕ and σ_{phi} inferred by encoder network and the whole process to obtain z is next:

$$z = \mu_\phi + \sigma_\phi \epsilon, \epsilon \sim N(0, 1) \tag{3.4}$$

Now let's denote z as z_0, $q_\phi(z|x)$ as $q_0(z_0|x)$ and apply described above transformation to it. We get some variable z_k with density $q_K(z_K|x) = q_0(z_0|x) \prod_{i=1}^{N} |det \frac{df_k}{dz_{k-1}}|^{-1}$. Now let's write down KL divergence for plain VAE:

$$\begin{gathered}\mathbb{D}_{\mathbb{KL}}(q_\phi(z|x)||p(z)) = \mathbb{E}_{z\sim q_\phi(z|x)} log(\frac{q_\phi(z|x)}{p(z} = \\ \mathbb{E}_{z\sim q_\phi(z|x)}(logq_\phi(z|x) - logp(z))\end{gathered} \tag{3.5}$$

Having new density for z_K we can rewrite it as follows:

$$\begin{aligned}\mathbb{D}_{\mathbb{KL}}(q_{K,\phi}(z_K|x)||p(z)) &= \mathbb{E}_{z_K\sim q_{K,\phi}(z_K|x)}(logq_{K,\phi}(z_K|x) - logp(z_K)) = \\ &\mathbb{E}_{z\sim q,\phi(z|x)}(logq_0(z_0|x) \prod_{i=1}^{N} |det \frac{df_k}{dz_{k-1}}|^{-1} - logp(z_K)) = \\ &\mathbb{E}_{z\sim q,\phi(z|x)}(logq_0(z_0|x) + \sum_{i=1}^{N} log|det \frac{df_k}{dz_{k-1}}|^{-1} - logp(z_K)) = \\ &\mathbb{E}_{z\sim q,\phi(z|x)}(logq_0(z_0|x) - \sum_{i=1}^{N} log|det \frac{df_k}{dz_{k-1}}| - logp(z_K))\end{aligned} \tag{3.6}$$

Now we have a tractable function, which we can optimize like a loss for plain VAE.

3.3 Planar Flow

Now let's explore simple example of such flow with planar transformations. Let's define f as follows:

$$f(z) = z + uh(w^T z + b), \tag{3.7}$$

where $u, w \in R^n$, $b \in R$, h is a differentiable element-wise non-linearity. We can compute determinant for this transformation as follows:

$$\psi(z) = h^{-1}(w^T z + b)w|det \frac{df}{dz}| = |det(I + u\psi(z)^T)| = |1 + u^T \psi(z)| \tag{3.8}$$

This is an extremely simple instance of transformation suitable for such flows because it only contracts and expands space of latent perpendicular to the hyperplane $w^T z + b = 0$.

3.4 Inverse Autoregressive Flow

In (Kingma et al., 2016) another type of flow based on an inverse transformation was proposed. Let $z \in R^n$ with components $\{z_i\}_{i\in\{1,\dots,n\}}$ and $\epsilon \sim N(0, 1), \epsilon \in R^n$ $\mu_i : R^{i-1} \to R^{i-1}$ and $\sigma_i : R^{i-1} \to R^{i-1}$ are functions, which we use to infer mean

and variance of z_i based on previous $i-1$ components:

$$\begin{aligned} z_0 &= \mu_0 + \sigma_0 \epsilon_0 \\ z_i &= \mu_i(z_{1:i-1}) + \sigma_i(z_{1:i-1})\epsilon_i \end{aligned} \tag{3.9}$$

Obviously, $\frac{d\mu_i}{dz_j} = 0$ and $\frac{d\sigma_i}{dz_j} = 0$ for each $j \geq i$, so Jacobian of such transformation is lower triangular with σ_i on the diagonal. However, the sampling process from this model is sequential and requires much time, so we can't efficiently draw samples from posterior approximated by such flow. But what we can do is to introduce inverse flow:

$$\epsilon_i = \frac{z_i - \mu_i(z_{1:i-1})}{\sigma_i(z_{1:i-1})} \tag{3.10}$$

We can denote μ as $\mu = [\mu_1, ..., \mu_n]$ and σ as $\sigma = [\sigma_1, ..., \sigma_n]$, then we can vectorize previous equation (μ and σ are element-wise vector functions):

$$\epsilon = \frac{z - \mu(z)}{\sigma(z)} \tag{3.11}$$

Like in previous case we can easily compute Jacobian of such transformation as $\frac{d\mu_i}{dz_j} = 0$ and $\frac{d\sigma_i}{dz_j} = 0$ for each $j \geq i$, so that $\frac{d\epsilon}{z}$ is a lower triangular with $-\sigma_i$ on diagonal.

Now it's important to note that functions μ can be represented as simple linear MLP, then we can parallelize those computations as we initially have all z_i. However, we can incorporate approach from (Germain et al., 2015), which computes μ and σ in single pass preserving autoregressive property.

Now we can reparameterize transformation derived above for simplicity, so for k-th step 3.1 of the flow it will look as follows:

$$z_k = \mu_k + \sigma_k z_{k-1} \tag{3.12}$$

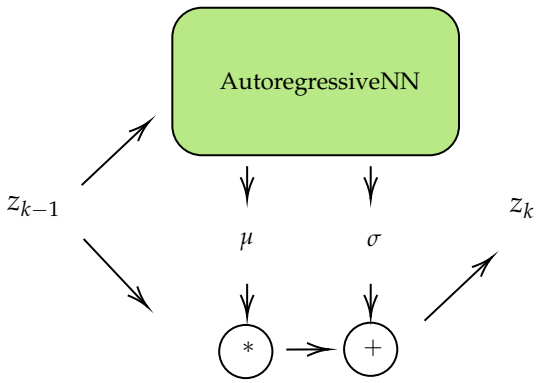

FIGURE 3.1: IAF step

It's also worth to notice that it looks like our initial transformation, but in this case we transform whole variable $z_{k-1} = [z_{k-1,1}, ..., z_{k-1,n}]$ into new variable $z_k = [z_{k,1}, ..., z_{k,n}]$ in one pass, and our initial transformation was defined for each component separately. But we still can compute Jacobian for this transformation in a similar way as $\frac{dz}{dz_{k-1}}$ is a lower triangular with $\sigma_{k,i}$ on diagonal. Finally, we can write

down complete density for the posterior:

$$q(z_K|x) = q_0(z_0|x) \prod_{k=1}^{K} \prod_{i=1}^{N} -\sigma_{k,i} \tag{3.13}$$

Interesting fact that as we model μ and σ as autoregressive autoencoder (e.g., MADE (Germain et al., 2015)) and compute forward transformation in one pass, we can easily stack a lot of such transformation and get enough modeling capacity to scale for high-dimensional variables.

This type of transformation used in a flow gives the ability to model a much more flexible posterior. Its efficiency had been shown on a broad range of image datasets. However, to our best knowledge, there are no reported results with this flow applied in a VAE for text modeling.

Chapter 4

Experiments

4.1 Datasets

We use two standard datasets in the field on which most of the results are reported - Yahoo and Yelp. Both of them contains millions of sentences with varying length, but we took a subset of only 100000 samples for training and 10000 for test and validation phase. The vocabulary size for Yahoo is 20001, and Yelp is 19997. In general, we used the same preprocessing strategy as in (He et al., 2019) and (Fang et al., 2019) to obtain comparable results.

drinks are weak and not made correctly . i guess that 's what you get with a barely legal bartender . do n't even bother ordering chicken strips unless you like paying $ 10 for 4 _UNK nuggets that are half burned . definitely wo n't be back .
square trade rip off warranties sold here ! i bought a product with a warranty that staples sells , it turns out it is by square trade . this is absolute garbage , i 've been on hold for 45 minutes just being transferred _UNK around india for my warranty that supposedly exists , what good is the warrant if i ca n't even talk to a person _UNK the call que keeps saying 1 minute to talk to someone, it 's been saying that for the last 45 minutes wtf !!!! rip off !! scam !

TABLE 4.1: Preprocessed samples from Yahoo dataset

if you were a character from a movie , who would it be and why ? it can be any movie , any movie character . morgan _UNK in " bruce almighty " who does n't want to be god ?
thanksgiving _UNK ? ? ? ? once the smell of a cooking turkey fills our house everyone keeps trying to sneak into the kitchen for bites of whatever is being cooked ... ..what does your family usually set out as _UNK & snacks (dont be afraid to share some recipes if you have any plz ! ! !) while the meals being cooked and to keep them out of the kitchen lol fresh veggies and dip is a favorite in my house , because it 's not too filling and it 's raw , which is a nice contrast to the dinner .

TABLE 4.2: Preprocessed samples from Yelp dataset

4.2 Evaluation

We consider two main metrics for evaluating the performance of our model:

- ELBO as a characteristic of the model's ability to match true data distribution alongside with perplexity.
- Value of KL divergence to understand if the model faced posterior collapse issue. However, how we'll show, higher KL divergence values itself doesn't mean that decoder uses latent to generate samples, so further discussion is needed.

4.3 Experiments Setup

We had the following set up for experiments:

- Embedding size of tokens - 512.
- Encoder - LSTM with hidden state size = 1024 and number of layers = 1.
- Decoder - LSTM with hidden state size = 1024 and number of layers = 1.
- Linear MLP to infer mean and std, latent size = 32.
- Latent z is used to initialize the hidden state of the decoder and also concatenated to word embeddings.
- 2 flow types - Planar and IAF with different number of steps (k=1,...,4). For IAF, we used one-layer MADE.
- We used SGD optimizer and applied learning rate decay by a factor of two each second epoch without loss improvement, and we used a batch size of 32 elements.
- For experiments with KL-annealing we used warm-up period before we started to increase KL term weight, initial KL weight was set to 0.1

4.4 Discussion

As Lagging Inference showed the best result from all related work, we choose it as a benchmark for our experimentation. Firstly, we reproduced results from this paper and as we can see from the table 4.3, we got similar results. Also, we experimented with annealing approach as it was useful in case of Lagging Inference.

Then we conducted experiments with planar flow. In (Bowman et al., 2015), authors mentioned that they tried to apply flows too, and it didn't help them, but they didn't specify which flow they used and didn't provide any metrics. However, the performance of a simple planar flow is too bad, and it doesn't help at all, so we can assume that they tried this type of flow. We can explain this model's performance by its nature: this flow contracts and expands variable around some hyperplane, so its expressive power is limited.

Next, we experimented with IAF. We thought that its expressive power would be sufficient to mitigate the posterior collapse problem; however, it helped only slightly in terms of KL divergence. It can be hard to make sense from those values, so we'll

Model	Yahoo			Yelp		
	ELBO	KL	PPL	ELBO	KL	PPL
VAE	329.0284	0.0031	61.5401	358.2366	0.0029	40.7037
VAE + Annealing	328.9191	0.0036	61.4559	358.5235	0.0058	40.8248
VAE + Lagging Inference	329.7644	5.673	62.1098	358.1073	3.7834	40.6493
VAE + Annealing + Lagging Inference	328.7168	6.2739	61.3004	359.0956	6.3059	41.0671
VAE + PF(k=1)	329.0186	0.0043	61.5325	358.5342	0.0048	40.8293
VAE + PF(k=2)	329.1406	0.0066	61.6266	359.0695	0.0052	41.056
VAE + PF(k=3)	329.0544	0.0057	61.5601	358.2151	0.0069	40.6947
VAE + PF(k=4)	329.0564	0.0081	61.5616	358.4898	0.0114	40.8105
VAE + IAF(k=1)	329.0461	0.0216	61.5537	358.5918	0.0163	40.8536
VAE + IAF(k=2)	329.1883	0.0285	61.6634	358.4321	0.0241	40.7862
VAE + IAF(k=3)	329.7845	0.0281	62.1254	359.0339	0.0194	41.0491
VAE + IAF(k=4)	329.9387	0.0293	62.2455	359.0856	0.0387	41.0629
VAE + IAF(4) + Annealing	333.983	0.019	65.4786	362.8214	0.0249	42.681
β-VAE (β=0.4)	329.5154	3.8941	61.9165	359.0339	2.4785	41.0408

TABLE 4.3: Experiments results

show how different values of KL divergence in different models affects the generational process to get more intuition about the performance of those models. In 4.5 we can see samples generated by plain VAE with collapsed posterior and VAE with the planar flow and slightly better, but still collapsed posterior. We can see that samples generated by those models are completely similar for different z. It means that the decoder ignores latent and deterministically generate sentences. But despite the low KL divergence IAF model has more variation in generated samples. Of course, it can't compete with Lagging Inference model, but we also need to remember that the Lagging model takes much more time to be trained. Also, we can see that KL divergence gradually increases as we stack more steps of the flow. Alongside that, ELBO stays the same. Taking into account slightly variable samples and ELBO similar to plain VAE model, we can say that this KL divergence increase is explained by better approximate posterior and not by insufficient optimization.

Then we checked samples from β-VAE to check our hypothesis that its high KL divergence (comparable to SA-VAE or Lagging Inference model) is due to bad optimization and posterior doesn't produce meaningful latent for the decoder. It appears that the quality of the samples produced by this model is awful. It shows that we can't rely solely on KL divergence values as they can be high because of bad optimization.

As IAF is designed to be scalable for high dimensional spaces, we also tried to increase the dimensionality of latent space. The results of those experiments can be found at 4.4. We found that KL divergence value increases as well as ELBO. Moreover, an increase in ELBO of the IAF model was more significant than in models with the planar flow or simple VAE. We think that it can be due to optimization problems as we use similar training procedures for every model, however high-dimensional IAF is much more complex than another two models. Unfortunately, we didn't have an opportunity to change our training procedure and try to optimize those models better to obtain more meaningful and comparable results, but it seems rewarding to do so in the future.

To sum up, those results suggest that our hypothesis about restricted approximate posterior as a source of a posterior collapse problem isn't correct. However, we saw that flows could slightly improve the performance of plain VAE and make latent drawn from posterior more informative for the decoder.

Model	Yahoo ELBO	Yahoo KL	Yahoo PPL	Yelp ELBO	Yelp KL	Yelp PPL
VAE(64)	330.4803	0.0026	62.669	359.9444	0.0027	41.4293
VAE(128)	333.5550	0.0052	65.1287	362.7560	0.0030	42.6522
VAE(64) + PF(4)	332.7247	0.002	64.4551	362.3535	0.0045	42.4749
VAE(128) + PF(4)	333.7192	0.0176	65.2627	362.8921	0.0085	42.7122
VAE(64) + IAF(4)	336.0405	0.0451	67.1874	364.1399	0.0364	43.2673
VAE(128) + IAF(4)	336.3144	0.065	67.4182	366.3162	0.0862	44.2525

TABLE 4.4: Experiments results (bigger latent size)

VAE
<s>what is the best way to get rid of hiccups ? hiccups are a spasm of the diaphragm . </s>
<s>what is the best way to get rid of hiccups ? hiccups are a spasm of the diaphragm . </s>
<s>what is the best way to get rid of hiccups ? hiccups are a spasm of the diaphragm . </s>
<s>what is the best way to get rid of hiccups ? hiccups are a spasm of the diaphragm . </s>
<s>what is the best way to get rid of hiccups ? hiccups are a spasm of the diaphragm . </s>
VAE + PF(4)
what is the best way to get rid of a headache ? i have a _UNK _UNK and i have a _UNK _UNK and
what is the best way to get rid of a headache ? i have a _UNK _UNK and i have a _UNK _UNK and
what is the best way to get rid of a headache ? i have a _UNK _UNK and i have a _UNK _UNK and
what is the best way to get rid of a headache ? i have a _UNK _UNK and i have a _UNK _UNK and
what is the best way to get rid of a headache ? i have a _UNK _UNK and i have a _UNK _UNK and
VAE + IAF(4)
<s>what is the best way to get a girl to like you ? ? ? ? ? ? ? ? ? ? ? ? </s>
<s>what is the best way to get rid of a headache ? i have been coughing for about 3 weeks now and i have been taking it for a while now . i have been taking it for about 2 weeks now and i have n't been able to get rid of it . any suggestions ?
<s>what is the best way to get rid of hiccups ? hiccups are a spasm of the diaphragm . the diaphragm is a spasm of the diaphragm . </s>
<s>what is the best way to get rid of hiccups ? hiccups are a spasm of the diaphragm and the diaphragm . </s>
<s>what is the best way to get rid of hiccups ? hiccups are a spasm of the diaphragm . the diaphragm is a spasm of the diaphragm . </s>
VAE + Lagging Inference

<s>what is the best way to get a free credit report on my credit report ? i need to know how to get a free credit report on my credit report , but i do n't know what to do . <s>what is the difference between a _UNK and a _UNK ? a _UNK is a _UNK , a _UNK , and a _UNK . </s>
<s>what is the best way to get rid of hiccups ? i have a friend who is a _UNK and i want to know if there is a way to get rid of it . i need to know how to get rid of it . please help ! !
<s>does anyone know where i can get a copy of the song " _UNK " by _UNK _UNK ? i think it 's called " _UNK " . </s>
<s>what is the meaning of the word " _UNK " ? it means " _UNK " . </s>
β-VAE, $\beta = 0.4$
<s>what is the difference between a _UNK and a _UNK ? a _UNK is a _UNK , _UNK , _UNK ,
<s>what is the difference between a _UNK and a _UNK ? a _UNK is a _UNK (_UNK) , _UNK
<s>what is the difference between a _UNK and a _UNK ? a _UNK is a _UNK , _UNK , _UNK ,
<s>what is the difference between a _UNK and a _UNK ? a _UNK is a _UNK , _UNK , _UNK ,
<s>what is the best way to get rid of hiccups ? hiccups are a spasm of the diaphragm and the diaphragm </s>

TABLE 4.5: Generated samples, Yahoo dataset

Chapter 5

Conclusions

5.1 Contribution

The main goal of this work was to address the posterior collapse issue with a better approximate posterior modeled by Normalizing Flows. According to this goal, we achieved the following results:

- We modeled variational posterior with two types of normalizing flows - planar and IAF. To our best knowledge, it's the first attempt to use IAF for text modeling. We reported results of our experiments on two standard benchmarks strictly following best practices in the field, so our results are comparable. We were unable to achieve state-of-the-art results presented in other papers and fully alleviate the posterior collapse problem, which suggests that restrictive approximate posterior isn't the main reason for the posterior collapse problem.
- However, we showed some evidence that a more flexible posterior can slightly increase KL divergence and make latent space more useful for the decoder. Also, we showed that KL divergence increases as a number of steps in a flow increases, but ELBO stays the same, suggesting that we can try longer chains to obtain even more flexible posteriors with higher KL divergence without harm to reconstruction capabilities. This also means that we can try to use this approach as additional improvement to state-of-the-art models (e.g., Lagging Inference).
- As a side result of our experiments, we showed that KL divergence value itself isn't a good metric, and models with high KL divergence values can face similar issues as collapsed models.

5.2 Future Work

There are a lot of possible ways to continue working in this direction. Here some possible minor improvements and more general ideas:

- Use additional metrics such as a number of active units and mutual information between generated data sample x and latent z to get a better understanding of model performance.
- Conduct experiments with longer chains of transformations. We saw that with each added step of the flow, KL divergence value increased, but ELBO stayed approximately the same. It suggests that we model can benefit from even more complex posterior.

- Combine approach based on approximated posterior by flows with Lagging Inference. More flexible posterior can benefit from aggressive updates.
- Continue experiments with high-dimensional z by changing training procedure to explore IAF capabilities in such scenario.
- Explore another type of flows. Flow-based approaches gained much attention in recent times, and a lot of new transformations were proposed, so we can experiment with them instead of IAF.
- Also, we can try to change encoder and decoder networks from LSTM to Transformer-like models.
- Another interesting approach is to replace the autoregressive LSTM decoder and use a purely flow model instead as it was used in (van den Oord et al., 2017).

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (Sept. 2014). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv e-prints*, arXiv:1409.0473, arXiv:1409.0473. arXiv: 1409.0473 [cs.CL].

Bowman, Samuel R. et al. (Nov. 2015). "Generating Sentences from a Continuous Space". In: *arXiv e-prints*, arXiv:1511.06349, arXiv:1511.06349. arXiv: 1511.06349 [cs.LG].

Cremer, Chris, Xuechen Li, and David Duvenaud (Jan. 2018). "Inference Suboptimality in Variational Autoencoders". In: *arXiv e-prints*, arXiv:1801.03558, arXiv:1801.03558. arXiv: 1801.03558 [cs.LG].

Devlin, Jacob et al. (Oct. 2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv e-prints*, arXiv:1810.04805, arXiv:1810.04805. arXiv: 1810.04805 [cs.CL].

Fang, Le et al. (Aug. 2019). "Implicit Deep Latent Variable Models for Text Generation". In: *arXiv e-prints*, arXiv:1908.11527, arXiv:1908.11527. arXiv: 1908.11527 [cs.LG].

Germain, Mathieu et al. (Feb. 2015). "MADE: Masked Autoencoder for Distribution Estimation". In: *arXiv e-prints*, arXiv:1502.03509, arXiv:1502.03509. arXiv: 1502.03509 [cs.LG].

He, Junxian et al. (Jan. 2019). "Lagging Inference Networks and Posterior Collapse in Variational Autoencoders". In: *arXiv e-prints*, arXiv:1901.05534, arXiv:1901.05534. arXiv: 1901.05534 [cs.LG].

Higgins, Irina et al. (2017). "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *ICLR*.

Jimenez Rezende, Danilo and Shakir Mohamed (May 2015). "Variational Inference with Normalizing Flows". In: *arXiv e-prints*, arXiv:1505.05770, arXiv:1505.05770. arXiv: 1505.05770 [stat.ML].

Kingma, Diederik P and Max Welling (Dec. 2013). "Auto-Encoding Variational Bayes". In: *arXiv e-prints*, arXiv:1312.6114, arXiv:1312.6114. arXiv: 1312.6114 [stat.ML].

Kingma, Diederik P. et al. (June 2016). "Improving Variational Inference with Inverse Autoregressive Flow". In: *arXiv e-prints*, arXiv:1606.04934, arXiv:1606.04934. arXiv: 1606.04934 [cs.LG].

Ott, Myle et al. (Apr. 2019). "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *arXiv e-prints*, arXiv:1904.01038, arXiv:1904.01038. arXiv: 1904.01038 [cs.CL].

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (Sept. 2014). "Sequence to Sequence Learning with Neural Networks". In: *arXiv e-prints*, arXiv:1409.3215, arXiv:1409.3215. arXiv: 1409.3215 [cs.CL].

van den Oord, Aaron et al. (Nov. 2017). "Parallel WaveNet: Fast High-Fidelity Speech Synthesis". In: *arXiv e-prints*, arXiv:1711.10433, arXiv:1711.10433. arXiv: 1711.10433 [cs.LG].

Vaswani, Ashish et al. (June 2017). "Attention Is All You Need". In: *arXiv e-prints*, arXiv:1706.03762, arXiv:1706.03762. arXiv: 1706.03762 [cs.CL].